7) Polymorphism — Compile time

↓

Means many forms.

Set of code behaves different in different situation

Two types

1. Compile time : Decide behaviour on compile time
2. Run time : Decide behaviour on run time

Compile Time

1) Achieved by Function Overloading & operator overloading

2) Function overloading

```
int fun(int a, int b){
}
int fun(int a){
}
int fun(){
}
int main(){
{
```

overloaded functions

```
void fun(){
}
```

we can't differentiate functions using return type compiler raises an ambiguity

* we can't differentiate function by return type

2) **operator overloading**

$2+3 = 5$ (Integer)

$3.5 + 1.5 = 5.0$ (double)

'a' + 'b' = ((char)

90 + 57 →

- One operator with different parameter.

~~II) we can~~ II) we can achieve the compile time polymorphism

~~method~~ function overriding

A → Base → print() → we have print() function in base class.

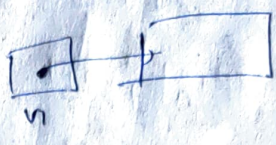
↓

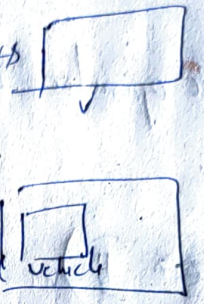
B → child → print() (The derived class also created the function named print)

This is called function overriding.

Program

```
int main() {
    Vehicle V;
    Car C;
    V.print()
    C.print()
    Vehicle *V1 = new Vehicle;
    Vehicle *V2;
    if V2 != &V
```





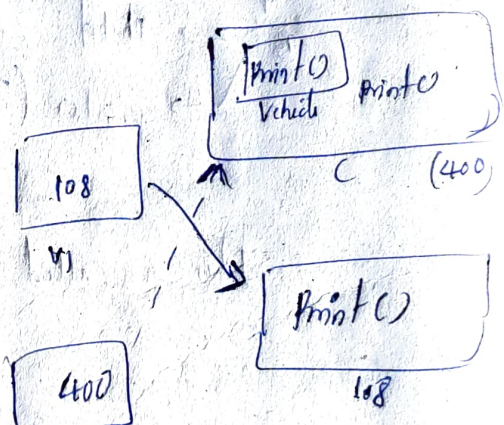
V2 = &C

(Base class pointer can point to derived class object)
But vice versa is not allowed.

⇒ V2 → print()

↓

Vehicle class print function is called



⇒ V2 → Pointer can access those properties of vehicle class

⇒ V2 → print() : This one calls vehicle class print function

↳ Compile time polymorphism.